Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

МІКРОКОНТРОЛЕРНА СИСТЕМА ВИЗНАЧЕННЯ МІСЦЕЗНАХОДЖЕННЯ В РЕАЛЬНОМУ ЧАСІ З ВИКОРИСТАННЯМ WEMOS D1 MINI І WI-FI

MICROCONTROLLER REAL-TIME LOCATION SYSTEM USING WEMOS D1 MINI AND WI-FI

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Кононович Микола Сергійович

(підпис)

Керівник:
к.т.н., доцент
Христинець Наталія Анатоліївна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» _____ 2025 р.
Гарант освітньої програми:
___
___

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: «*Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ к. т. н., доцент Тарас Терлецький
«_____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кононовичу Миколі Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Мікроконтролерна система визначення місцезнаходження в реальному часі з використанням Wemos D1 Mini і Wi-Fi*

Керівник роботи: *Христинець Наталія Анатоліївна, к.т.н., доцент*

затверджені наказом закладу вищої освіти від «__» ____ 202_ року _____

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи _____

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях по темі Інтернету речей, опубліковані роботи в даній області та інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Вступ.
Теоретичні відомості про мікроконтролерні системи
Аналіз об'єкту розробки
Розробка системи визначення місцезнаходження
Висновки.

5. Перелік графічного (ілюстративного) матеріалу:
Фото мікроконтролеру, схеми взаємодії компонентів системи, рисунки інтерфейсу користувача, лістинги коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні відомості про мікроконтролерні системи	*Христинець Н.А., доцент*		
Аналіз об'єкту розробки	*Христинець Н.А., доцент*		
Розробка системи визначення місцезнаходження	*Христинець Н.А., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., асистент*		

7. Дата видачі завдання __.__.____ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	09.02.2025 р.	Виконано
2.	*Вибір апаратної та програмної бази для проєкту*	28.02.2025 р.	Виконано
3.	*Проєктування та тестування системи визначення місцезнаходження*	25.03.2025 р.	Виконано
4.	*Висновки та пропозиції*	04.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	10.04.2025 р.	Виконано
6.	*Формування додатків*	01.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	05.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	10.05.2025 р.	Виконано
9.	*Нормоконтроль*	27.05.2025 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	04.06.2025 р.	Виконано
11.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	10.06.2025 р.	Виконано

Здобувач вищої освіти _____ (підпис) Кононович М.С. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Христинець Н.А. (прізвище, ініціали)

АНОТАЦІЯ

Кононович М.С. Мікроконтролерна система визначення місцезнаходження в реальному часі з використанням Wemos D1 Mini і Wi-Fi. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків.

Перший розділ присвячено аналізу сучасних мікроконтролерних систем, де розглянуто архітектурні особливості Wemos D1 Mini на базі ESP8266, принципи роботи GPS-модулів та бездротових технологій у контексті IoT. Досліджено переваги використання двох методів геолокації (браузерної API та апаратного GPS) для підвищення надійності системи.

Другий розділ містить порівняльний аналіз технічних рішень, де обґрунтовано вибір Wemos D1 Mini серед аналогічних платформ (Arduino Uno, STM32), проведено оцінку енергоефективності та продуктивності використаних бібліотек (TinyGPS++, BearSSL). Розкрито принципи реалізації захищеного HTTPS-сервера з TLS 1.2 та механізми оптимізації енергоспоживання.

Третій розділ описує практичну реалізацію системи, включаючи розробку адаптивного веб-інтерфейсу з використанням Leaflet.js, інтеграцію з OpenStreetMap API та створення прошивки для мікроконтролеру.

Ключові слова: Wemos D1 Mini, GPS NEO-6M, IoT, геолокація, Arduino IDE, TinyGPS++, HTTPS.

ANNOTATION

Kononovych M.S. Microcontroller-Based Real-Time Location Tracking System Using Wemos D1 Mini and Wi-Fi. Manuscript.

Bachelor’s Qualification Thesis, Educational Program "Computer Engineering", specialty 123 "Computer Engineering". Lutsk National Technical University, Lutsk, 2025.

The qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter is dedicated to the analysis of modern microcontroller systems. It discusses the architectural features of the Wemos D1 Mini based on the ESP8266, the operating principles of GPS modules, and wireless technologies in the context of IoT. The advantages of using two geolocation methods (browser API and hardware GPS) are explored to enhance the system's reliability.

The second chapter contains a comparative analysis of technical solutions, justifying the selection of the Wemos D1 Mini among similar platforms (Arduino Uno, STM32). It evaluates the energy efficiency and performance of the utilized libraries (TinyGPS++, BearSSL) and describes the implementation of a secure HTTPS server with TLS 1.2 and energy consumption optimization mechanisms.

The third chapter describes the practical implementation of the system, including the development of an adaptive web interface using Leaflet.js, integration with the OpenStreetMap API, and firmware creation for the microcontroller.

Keywords: Wemos D1 Mini, GPS NEO-6M, IoT, geolocation, Arduino IDE, TinyGPS++, HTTPS.

ЗМІСТ

ВСТУП

Актуальність теми. Розвиток IoT-технологій та зростаюча потреба в точному позиціюванні об'єктів визначають необхідність створення енергоефективних геолокаційних систем. Сучасні рішення потребують інтеграції бездротових технологій, захищених комунікаційних протоколів та адаптивних інтерфейсів, що особливо актуально для застосувань у логістиці, навігації та системах моніторингу.

Метою роботи є створення мікроконтролерної системи визначення місцезнаходження в реальному часі з використанням Wemos D1 Mini і Wi-Fi.

Об'єкт дослідження – процеси визначення місцезнаходження в мікроконтролерних системах.

Предмет дослідження – програмно-апаратна реалізація геолокаційної системи на базі Wemos D1 Mini з інтеграцією бездротових технологій.

Завдання, які необхідно виконати:

– дослідити основні теоретичні відомості про мікроконтролерні системи;

– спроектувати архітектуру клієнт-серверної взаємодії;

– реалізувати двохрежимний механізм геолокації з автоматичним переключенням між браузерним API та апаратним GPS;

– розробити захищений веб-інтерфейс з підтримкою HTTPS/TLS 1.2 та REST API;

– візуалізувати геопросторові дані з використанням Leaflet.js та OpenStreetMap.

Структура й обсяг роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел із 21 найменування. Повний обсяг роботи становить – 43 сторінки, 25 рисунків та 5 таблиць.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО МІКРОКОНТРОЛЕРНІ СИСТЕМИ

1.1 Основні поняття про мікроконтролери

Мікроконтролер – це спеціалізована інтегральна схема, призначена для виконання конкретних завдань у вбудованих системах. Його архітектура поєднує центральний процесор, пам’ять та периферійні пристрої введення/виведення на одному кристалі, що забезпечує ефективне управління електронними системами без необхідності використання додаткових обчислювальних ресурсів.

Основними компонентами мікроконтролера є центральний процесор (CPU), пам’ять та периферійні модулі (рис 1.1).

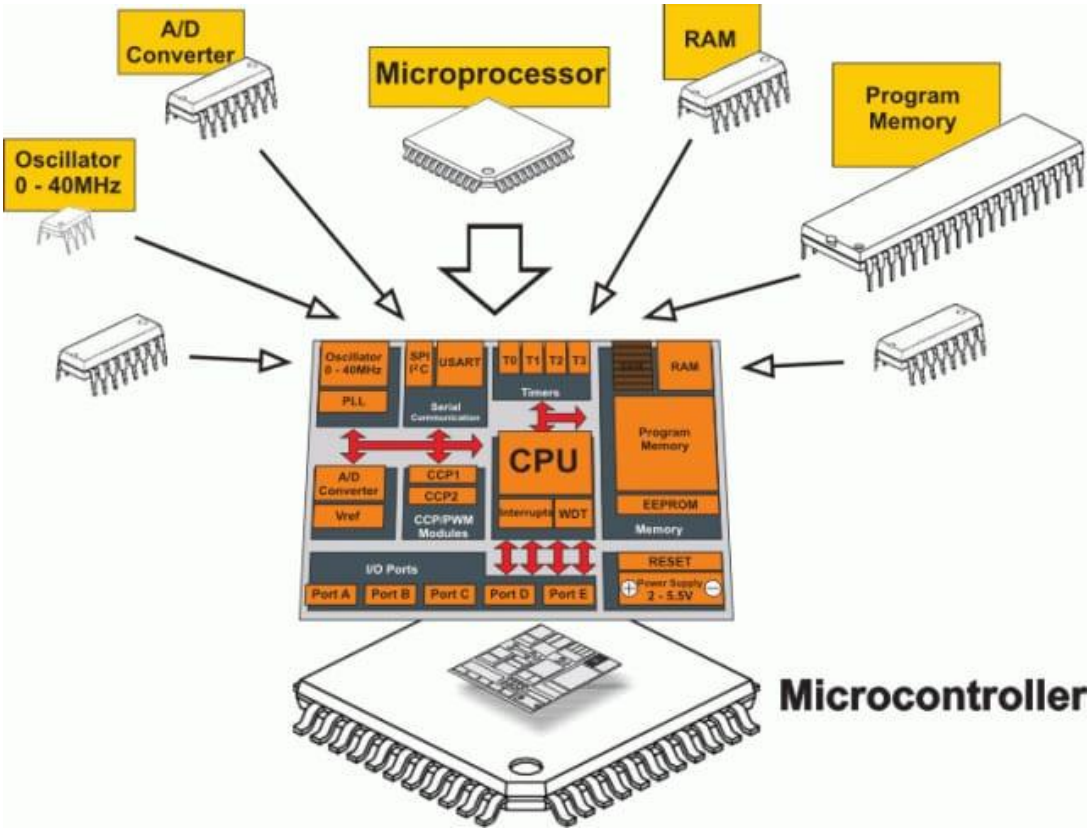

Рисунок 1.1 – Складові мікроконтролеру [1]

Центральний процесор виконує арифметичні та логічні операції, керує передачею даних та забезпечує взаємодію між складовими пристрою. Пам’ять мікроконтролера поділяється на програмну (ROM, Flash), яка містить прошивку

з інструкціями, та оперативну (RAM), що використовується для тимчасового зберігання даних під час виконання програми. Периферійні пристрої включають аналого-цифрові перетворювачі (ADC), цифрові інтерфейси зв’язку (UART, SPI, I2C), таймери, широтно-імпульсну модуляцію (PWM) та інші функціональні блоки, необхідні для інтеграції мікроконтролера у складні електронні системи [2].

Окрім процесора, пам’яті та периферійних пристроїв введення-виведення, які є визначальними елементами мікропроцесора, існують і додаткові компоненти, що часто інтегруються до його складу. Периферійні пристрої введення-виведення являють собою допоміжні елементи, що забезпечують взаємодію між пам’яттю та процесором. До цієї категорії належать різні компоненти, що виконують допоміжні функції. Наявність периферійних пристроїв є необхідною умовою функціонування мікропроцесора, оскільки саме вони забезпечують його зв’язок із зовнішнім середовищем.

До інших допоміжних компонентів мікроконтролера належать:

– системна шина, що функціонує як комунікаційний канал, який об’єднує всі компоненти мікроконтролера, забезпечуючи їхню взаємодію (рис. 1.2);

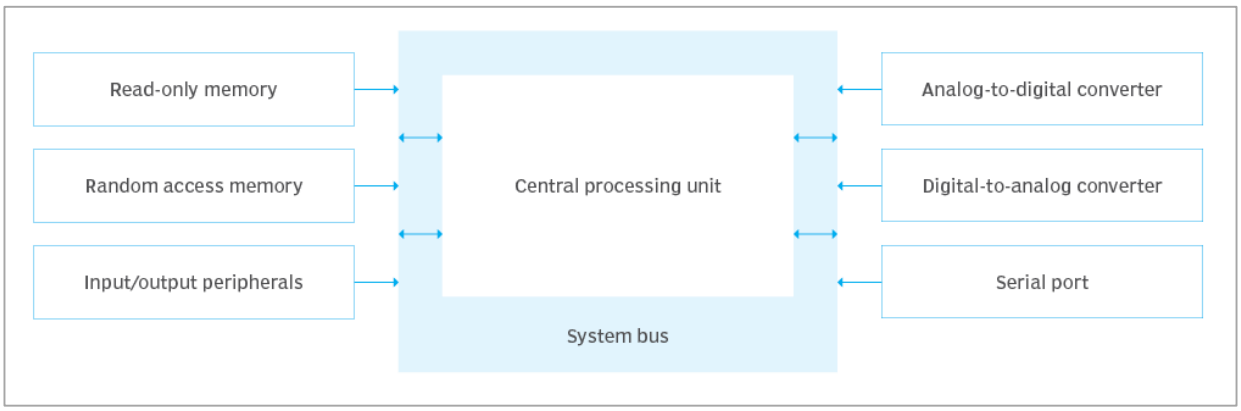

Рисунок 1.2 – Підключення компонентів до шини [3]

– аналого-цифровий перетворювач (АЦП), що здійснює перетворення аналогових сигналів у цифрові. Це дає змогу процесору мікроконтролера взаємодіяти із зовнішніми аналоговими пристроями, зокрема датчиками;

– цифро-аналоговий перетворювач (ЦАП), що виконує функцію, обернену до АЦП, забезпечуючи передачу цифрових сигналів мікроконтролера до зовнішніх аналогових пристроїв;

– послідовний порт, який є одним із типів портів введення-виведення, що забезпечує з’єднання мікроконтролера із зовнішніми пристроями. Його функціональність подібна до USB або паралельного порту, проте метод обміну даними відрізняється.

Функціонування мікроконтролера ґрунтується на обробці вхідних сигналів через периферійні пристрої, збереженні тимчасових даних у пам’яті та виконанні команд, закладених у програмному забезпеченні. Отримані дані аналізуються центральним процесором, після чого мікроконтролер здійснює відповідні керувальні дії через вихідні інтерфейси. Завдяки компактності та енергоефективності мікроконтролери широко використовуються в різних галузях, включаючи промислову автоматику, побутову електроніку, медичне обладнання та інтернет речей [4].

Залежно від архітектури мікроконтролери можуть базуватися на фон-Нейманівській або Гарвардській моделі. Фон-Нейманівська архітектура використовує єдину шину для обміну інструкціями та даними, що спрощує розробку, але обмежує продуктивність. Гарвардська архітектура передбачає розділення потоків команд і даних, що забезпечує вищу швидкість обробки інформації. Крім того, мікроконтролери класифікуються за розрядністю процесора: 8-бітові пристрої є енергоефективними та використовуються у простих системах, 16-бітові та 32-бітові забезпечують вищу продуктивність для складних обчислювальних завдань.

Мікроконтролери програмуються на різних мовах, серед яких найбільш поширеними є C, C++ та Python. Їхнє використання дозволяє реалізовувати як базові алгоритми керування, так і складні системи аналізу даних [5]. Сучасні мікроконтролери підтримують численні периферійні модулі, що розширює їхні можливості та дозволяє інтегрувати у широкий спектр застосувань. Завдяки високому рівню гнучкості, продуктивності та низькому енергоспоживанню,

мікроконтролери є основою розробки інтелектуальних пристроїв та автоматизованих систем.

1.2 Огляд Wemos D1 Mini

WeMos D1 Mini – це компактна мікроконтролерна платформа, що базується на чипі ESP8266, який поєднує можливості мікропроцесорного керування та бездротового зв’язку за стандартом Wi-Fi (рис 1.3).

Рисунок 1.3 – Зовнішній вигляд Wemos D1 Mini [6]

Завдяки своїм технічним характеристикам та мініатюрним розмірам, ця плата широко використовується у розробці пристроїв Інтернету речей (IoT), автоматизації та прототипуванні електронних систем.

WeMos D1 Mini забезпечує 18 цифрових входів/виходів, з яких чотири можуть функціонувати як виходи широтно-імпульсної модуляції (PWM). Крім того, плата оснащена одним аналоговим входом, який підтримує 10-бітне перетворення сигналу. Інтерфейси UART, I2C та SPI дають змогу реалізовувати комунікацію з іншими периферійними пристроями, що розширює можливості її

застосування [7]. Наявність вбудованого USB-to-serial перетворювача на базі чипа CH340G спрощує процес прошивки та налагодження.

Плата підтримує кілька режимів роботи Wi-Fi: точку доступу (AP), станцію (STA) та комбінований режим (AP+STA). Це дозволяє використовувати WeMos D1 Mini як самостійну керуючу одиницю або як частину складної мережевої системи. Вбудований мікроконтролер працює на базі архітектури RISC із тактовою частотою 80 МГц, з можливістю підвищення до 160 МГц, що забезпечує ефективну обробку даних у реальному часі. Об’єм флеш-пам’яті становить 4 МБ, що дозволяє зберігати програмний код та налаштування для автономної роботи пристрою.

WeMos D1 Mini може працювати від живлення 3,3 В або 5 В (через USB-інтерфейс), що забезпечує сумісність із широким спектром периферійних модулів. Крім стандартної версії, доступна модифікація D1 Mini Pro, яка включає покращений модуль зв’язку та схему заряду акумулятора. Завдяки підтримці середовищ програмування Arduino IDE і MicroPython, ця плата є гнучким інструментом для реалізації різноманітних інженерних завдань.

Arduino IDE – це офіційне програмне середовище для написання, редагування, компіляції та завантаження коду на мікроконтролери Arduino. Воно створене для того, щоб спростити розробку програмного забезпечення для апаратних платформ Arduino, навіть для новачків у програмуванні та електроніці [8].

MicroPython – це спрощена та ефективна реалізація мови програмування Python 3, яка містить невелику підмножину стандартної бібліотеки Python і оптимізована для роботи на мікроконтролерах та в обмежених середовищах [9].

Серед додаткових можливостей слід відзначити наявність системної шини, яка забезпечує взаємодію всіх компонентів, а також індикаторного світлодіода, підключеного до GPIO2 (D4), що дозволяє здійснювати діагностику роботи пристрою. Модульний форм-фактор та розміщення роз’ємів у стандартному сітковому форматі (2,54 мм) забезпечують сумісність з макетними платами, що спрощує процес прототипування.

Загалом, WeMos D1 Mini є високоефективною платформою для реалізації проєктів у сфері автоматизації та IoT. Її компактність, розширені комунікаційні можливості та підтримка різних середовищ програмування роблять її зручним інструментом для інженерів та розробників електронних систем.

1.3 Порівняльний аналіз засобів розробки

Вибір технологічного стеку для реалізації системи визначення місцезнаходження ґрунтується на аналізі енергоефективності, апаратної сумісності та можливості обробки геопросторових даних у реальному часі. Базовим компонентом системи є мікроконтролер Wemos D1 Mini, який забезпечує оптимальний баланс між продуктивністю та споживанням енергії завдяки архітектурі ESP8266 з інтегрованим WiFi-модулем. Порівняно з аналогічними платформами типу Arduino Uno, який обмежений 32 КБ флеш-пам'яті та 2 КБ ОЗП із відсутністю вбудованого бездротового інтерфейсу, даний пристрій пропонує ширші комунікаційні можливості та більший обсяг флеш-пам'яті (4 МБ), що є критичним фактором для зберігання та обробки GPS-даних. Використання плат на базі STM32 потребувало б додаткових модулів для реалізації WiFi-зв'язку, що збільшувало б енергоспоживання та складність проєкту [10]. Порівняння характеристик цих плат описано в таблиці 1.1.

Таблиця 1.1 – Порівняння популярних плат для IoT

Характеристика	Wemos D1 Mini (ESP8266)	Arduino Uno (ATmega328P)	STM32-based Boards
Архітектура процесора	RISC (Tensilica L106)	AVR 8-bit RISC	ARM Cortex-M (32-bit)
Тактова частота	80–160 MHz	16 MHz	72–216 MHz (залежно від моделі)
Флеш-пам'ять	4 MB	32 KB	64 KB – 2 MB
ОЗП (RAM)	80 KB	2 KB	20–512 KB
Бездротовий інтерфейс	Вбудований Wi-Fi (802.11 b/g/n)	Відсутній	Вимагає зовнішнього модуля
Енергоспоживання	~80 mA (активний режим)	~25 mA (активний режим)	~36–50 mA
GPIO кількість	11 цифрових + 1 аналоговий	14 цифрових + 6 аналогових	16–100+ (залежно від моделі)

Продовження таблиці 1.1

Інтерфейси зв'язку	UART, SPI, I2C	UART, SPI, I2C	UART, SPI, I2C, USB, CAN
Ціна	~$3–5	~$20–25	~$8–15 (базові моделі)
Розрядність	32-bit	8-bit	32-bit
АЦП роздільна здатність	10-bit	10-bit	12-bit
Підтримка RTOS	Так (FreeRTOS)	Обмежена	Так (FreeRTOS, Zephyr тощо)
Програмне середовище	Arduino IDE, MicroPython	Arduino IDE	Arduino IDE, STM32CubeIDE
Критичні переваги	Інтегрований Wi-Fi, велика пам’ять	Простота використання	Висока продуктивність
Недоліки	Обмежена кількість аналогових входів	Малі ресурси пам’яті	Вища складність розробки

Використання бібліотеки TinyGPS++ дозволяє ефективно парсити NMEA-рядки з GPS-модуля, забезпечуючи точність визначення координат у межах 2.5 метрів. Альтернативне рішення, як наприклад Adafruit GPS Library потребує більших апаратних ресурсів, що робить його менш прийнятним для систем з обмеженим обсягом оперативної пам'яті [11]. Порівняння цих бібліотек описано в таблиці 1.2

Таблиця 1.2 – Порівняння бібліотек для роботи з GPS

Характеристика	TinyGPS++	Adafruit GPS Library
Тип	Легка бібліотека для парсингу NMEA-рядків	Повнофункціональна бібліотека для роботи з GPS-модулями
Споживання оперативної пам’яті	Низьке – оптимізована для мікроконтролерів з малою ОЗП	Високе – потребує більше ресурсів
Продуктивність парсингу	Висока – ефективне оброблення NMEA	Нижча через додаткову обробку і функціонал
Додаткові функції	Координати, висота, швидкість, час, курс	Широкий набір функцій: стан супутників, якість сигналу, логування
Сумісність	Arduino, ESP8266/ESP32, STM32 тощо	Arduino, деякі сумісні плати ESP
Простота використання	Відносно проста	Вимагає більше налаштувань
Підходить для систем з обмеженим ОЗП	Так	Ні

Аналіз роботи антени показує, що використання пасивної керамічної антени з коефіцієнтом посилення 3dB забезпечує стабільний прийом

супутникових сигналів у міських умовах, хоча й поступається активним антенам з зовнішнім підсилювачем у зонах зі складною електромагнітною обстановкою [12]. Повне порівняння цих двох типів антен описане у таблиці 1.3.

Таблиця 1.3 – Порівняння типів антен

Характеристика	Пасивна керамічна антена	Активна GPS антена
Тип	Без підсилювача	З вбудованим підсилювачем
Коефіцієнт підсилення	~3 dB	20–30 dB (залежно від моделі)
Потреба в живленні	Ні	Так (зазвичай 3.3–5 В через антений вхід або окремо)
Розміри	Компактна (зазвичай 9×9 мм, 12×12 мм)	Більші (через корпус та підсилювач)
Прийом у міських умовах	Стабільний, але чутливий до розташування	Надійний, краща чутливість до слабких сигналів
Прийом у зашумлених/складних умовах	Слабший	Кращий завдяки активному підсиленню
Ціна	Низька (~$1–2)	Вища (~$5–10)
Складність інтеграції	Проста – підключення без живлення	Потрібне забезпечення живлення для підсилювача
Енергоспоживання	Відсутнє	Помірне (~10–20 мА, залежно від моделі)
Ідеальне застосування	Компактні, енергоефективні системи	Системи з потребою у високій якості сигналу

Інтеграція програмно-апаратних компонентів реалізована через UART-інтерфейс, що забезпечує швидкість передачі даних до 115200 бод [13]. Порівняно з I2C та SPI протоколами, обраний метод комунікації демонструє кращу стійкість до перешкод на великих відстанях між сенсором та мікроконтролером (табл. 1.4). Вибір середовища розробки Arduino IDE обумовлений підтримкою кросплатформенності та широкою бібліотечною базою, що значно скорочує час написання прошивки порівняно з низькорівневим програмуванням на мові C.

Таблиця 1.4 – Порівняння інтерфейсів

Характеристика	UART	I2C	SPI
Кількість проводів	2 (TX, RX)	2 (SDA, SCL)	4 (MISO, MOSI, SCK, SS)
Максимальна швидкість	До ~1 Мбіт/с (типово 115200 бод)	До 400 кбіт/с (стандарт), 3.4 Мбіт/с (HS)	До 10–20 Мбіт/с
Дальність зв’язку	Вища, особливо на низьких швидкостях	Слабша через вразливість до шуму	Коротка – чутливий до довжини ліній

Продовження таблиці 1.4

Чутливість до шуму	Низька (висока стійкість)	Висока	Середня
Підтримка адресації	Точка-точка (1:1)	Так (до 127 пристроїв)	Ні, керування вручну через CS
Складність реалізації	Проста	Середня	Складніша (більше ліній, контроль)
Ідеальне застосування	Довгі кабелі, GPS, модеми	Сенсори, EEPROM	Швидкі сенсори, дисплеї

РОЗДІЛ 2
АНАЛІЗ ОБ'ЄКТУ РОЗРОБКИ

2.1 Опис засобів вирішення поставленого завдання

Система визначення місцезнаходження реалізована на базі двох взаємодоповнюючих методів геолокації, що забезпечують надійність та енергоефективність роботи. Первинним механізмом є використання браузерної Geolocation API, яка активується при завантаженні веб-інтерфейсу з вбудованого HTTPS-сервера мікроконтролера (для роботи Geolocation API, сервер повинен мати підписаний SSL сертифікат).

Geolocation API – це веб-інтерфейс, який дозволяє веб-сайтам отримувати географічне розташування пристрою користувача за його згодою [14]. Він є частиною стандартів HTML5 і підтримується більшістю сучасних браузерів.

Мікроконтролер Wemos D1 Mini функціонує як захищений веб-сервер з підтримкою HTTPS-протоколу. Використання вбудованого TLS/SSL-стеку з самопідписаним сертифікатом дозволяє забезпечити шифрований обмін даними між клієнтом та сервером, виключаючи можливість перехоплення конфіденційної інформації, та забезпечуючи роботу браузерного Geolocation API. Апаратна складова системи включає GPS-модуль NEO-6M з зовнішньою антеною (рис. 2.1), підключений через асинхронний послідовний інтерфейс UART з використанням бібліотеки TinyGPS++ для аналізу NMEA-телекомунікаційних протоколів.

Рисунок 2.1 – GPS-модуль NEO-6M з зовнішньою антеною [15]

Клієнтський інтерфейс представлений у вигляді адаптивної веб-сторінки, розробленої з використанням стандартів HTML5 та CSS3, що забезпечує крос-платформенну сумісність. Інтерактивна мапа на базі бібліотеки Leaflet.js (рис. 2.2) інтегрована через JavaScript-API, яке реалізує двоетапний механізм запиту геоданих [16].

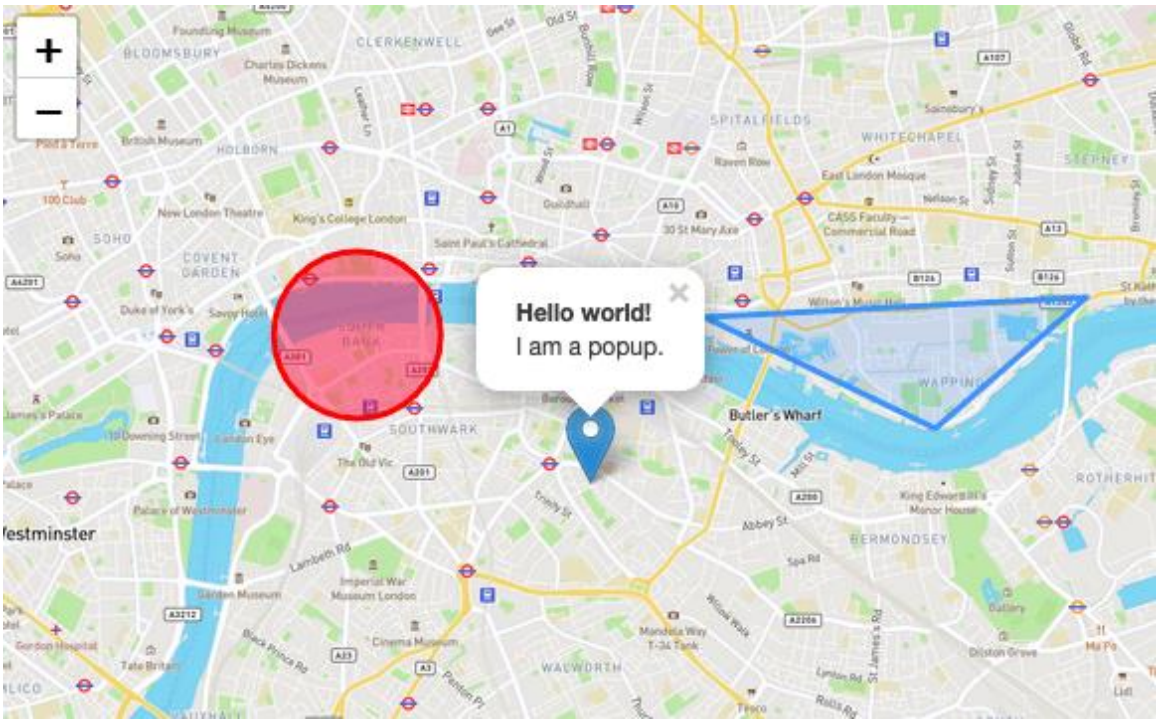

Рисунок 2.2 – Базовий інтерфейс мапи з Leaflet.js

Серверна частина системи представлена у вигляді RESTful API з ендпоїнтом /api/coordinates, який надає дані у форматі JSON з інформацією про широту, довготу та точність позиціювання, у разі роботи через апартний GPS-модуль. Взаємодія між апаратними та програмними компонентами забезпечується через механізм преривань та подій, де подія відмови в доступі до геолокації браузера ініціює перехід на апаратний режим збору даних.

RESTful API – це архітектурний стиль для створення веб-сервісів, що дозволяє різним системам обмінюватися даними через інтернет за допомогою стандартних HTTP-запитів (рис. 2.3). RESTful API забезпечує простий, уніфікований інтерфейс для взаємодії між клієнтом і сервером, що робить його популярним вибором для розробки веб-додатків і мобільних сервісів.

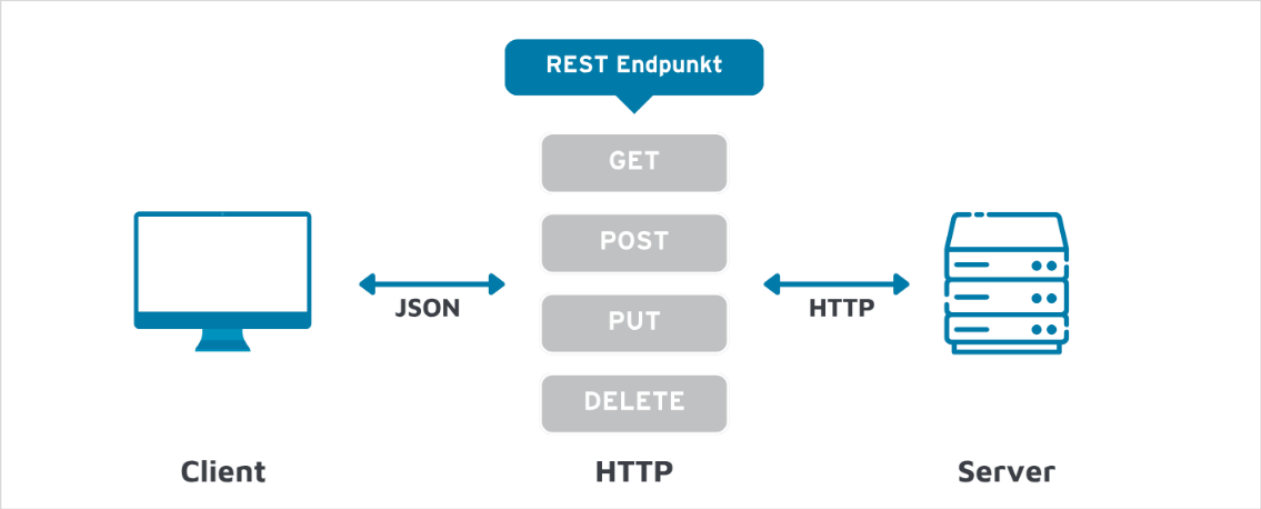

Рисунок 2.3 – Принцип роботи RESTful API [17]

Основними принципами REST є клієнт-серверна архітектура, безстанова взаємодія, кешування, уніфікований інтерфейс, багаторівнева система та можливість виконання коду на вимогу. Клієнт-серверна модель передбачає чітке розділення обов'язків між клієнтом, який ініціює запити, та сервером, який обробляє ці запити та повертає відповідні відповіді. Безстановість означає, що кожен запит містить всю необхідну інформацію для його обробки, і сервер не зберігає контекст між запитами, що сприяє покращенню масштабованості та надійності системи. Кешування дозволяє зменшити навантаження на сервер та покращити продуктивність, зберігаючи відповіді на попередні запити. Уніфікований інтерфейс забезпечує стандартизовану взаємодію між компонентами системи, використовуючи стандартні HTTP-методи (GET, POST, PUT, DELETE) для маніпуляції ресурсами. Багаторівнева система дозволяє розділити функціональність на окремі шари, що сприяє гнучкості та масштабованості архітектури. Можливість виконання коду на вимогу дозволяє серверам передавати клієнтам виконуваний код для розширення функціональності.

Переваги RESTful API включають простоту реалізації, гнучкість, масштабованість та широке прийняття в індустрії. Завдяки використанню стандартних протоколів та форматів даних (наприклад, JSON або XML), RESTful API забезпечує ефективну та уніфіковану взаємодію між різними компонентами

системи. Безстанова природа RESTful API сприяє легкому масштабуванню та підвищенню надійності, оскільки кожен запит обробляється незалежно. Крім того, RESTful API підтримується більшістю сучасних веб-фреймворків та бібліотек, що полегшує його впровадження та використання в різноманітних програмних рішеннях.

У порівнянні з іншими технологіями, такими як SOAP або GraphQL, RESTful API пропонує більш легкий та гнучкий підхід до розробки веб-сервісів (табл. 2.1). SOAP використовує складніші протоколи та формати, що може ускладнювати реалізацію та інтеграцію. GraphQL, з іншого боку, надає клієнтам можливість запитувати лише необхідні дані, але може вимагати більш складної реалізації на стороні сервера [18]. RESTful API забезпечує баланс між простотою, гнучкістю та ефективністю, що робить його популярним вибором для багатьох розробників та організацій.

Таблиця 2.1 – Порівняння технологій передачі даних

Технологія	RESTful API	SOAP	GraphQL
Протокол	HTTP	HTTP, SMTP, TCP	HTTP
Формат даних	JSON, XML	XML	JSON
Становість	Безстанова	Станова (може зберігати стан)	Безстанова
Гнучкість запитів	Фіксовані кінцеві точки	Фіксовані операції	Динамічні запити
Складність реалізації	Низька	Висока	Середня
Підтримка кешування	Так	Обмежена	Обмежена
Підтримка стандартів	Відсутня	Висока (WS-Security, WS-ReliableMessaging)	Відсутня
Підтримка браузерами	Висока	Низька	Висока
Підходить для	Веб-додатків, мобільних додатків	Корпоративних систем, що потребують високої безпеки	Клієнтів з потребою в гнучких запитах

2.2 Проектування структури системи

Підхід на основі двох взаємодоповнюючих методів геолокації дозволяє отримувати координати напряму з пристрою користувача, який з великою вірогідністю має кращий вбудований GPS-модуль, що може працювати

практично в будь-якому середовищі та забезпечує фактично моментальне визначення місцезнаходження, а також дозволяє не задіювати апаратний GPS-модуль, що підключений до мікроконтролера, і який має обмеження по середовищу роботи, так як працює в основному лише під відкритим небом. Такий підхід дуже позитивно відображається на енергоспоживанні.

Загалом, таке архітектурне рішення передбачає наступний потік дій: спочатку користувач отримує доступ до веб-інтерфейсу через захищене HTTPS-з'єднання, після чого браузер автоматично ініціює запит геолокації через стандартний API (рис. 2.4). При отриманні дозволу дані передаються напряму з клієнтського пристрою, а у разі відмови система автоматично перемикається на апаратний GPS-модуль.

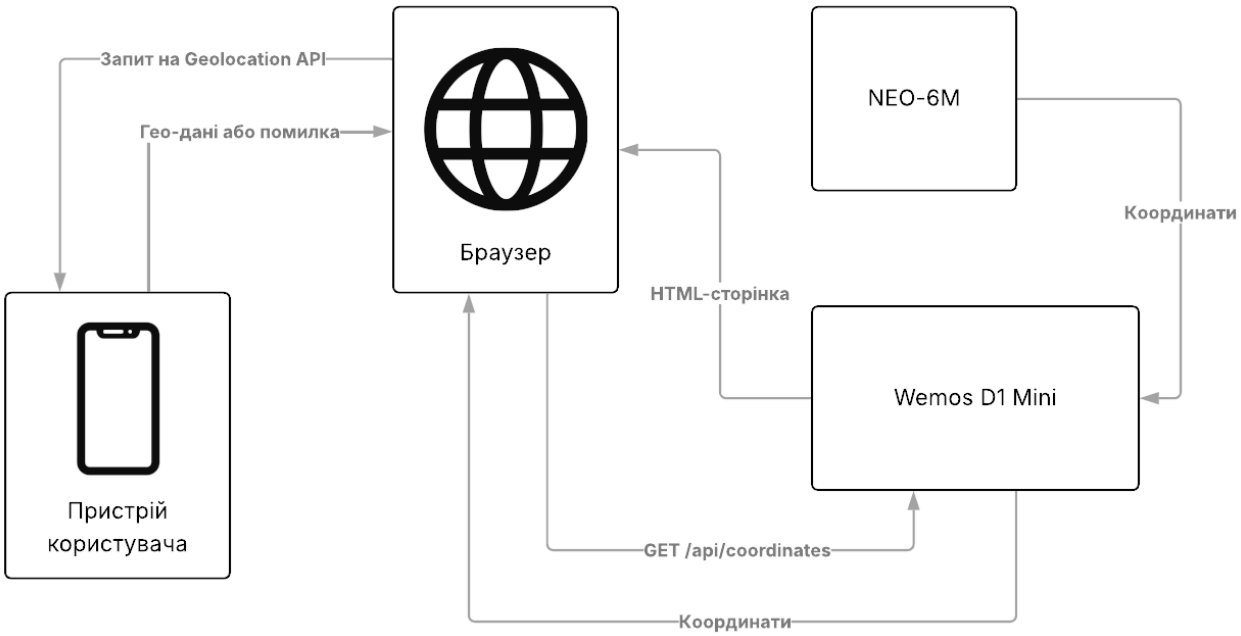

Рисунок 2.4 – Схема потоку дій при роботі системи

Антена NEO-6M підключена через UART-інтерфейс зі швидкістю 9600 бод. Мікроконтролер постійно опитує GPS-приймач, декодуючи NMEA-рядки за допомогою бібліотеки TinyGPS++, що забезпечує аналіз сигналів від 14 супутників одночасно.

2.3 Принципи взаємодії між компонентами системи

Архітектура системи ґрунтується на клієнт-серверній моделі з двонаправленим обміном даними через захищені мережеві протоколи. Мікроконтролер Wemos D1 Mini функціонує як HTTPS-сервер з реалізацією TLS 1.2 за допомогою вбудованої криптобібліотеки BearSSL, що забезпечує аутентифікацію на основі самопідписаного X.509 сертифіката (стандарт RSA-2048 з хешуванням SHA256) [19].

HTTPS (HyperText Transfer Protocol Secure) – це мережевий протокол прикладного рівня, який забезпечує захищений обмін даними між веб-клієнтом (наприклад, браузером) і сервером. На відміну від звичайного HTTP, який передає інформацію у відкритому вигляді, HTTPS використовує криптографічний протокол TLS (Transport Layer Security) для шифрування трафіку, що унеможливлює несанкціоноване перехоплення, модифікацію або підміну даних під час їх передавання. Таким чином, HTTPS гарантує конфіденційність, цілісність і автентичність інформаційного обміну в мережі.

TLS (Transport Layer Security) – це криптографічний протокол, який слугує для забезпечення захищеного зв’язку між двома сторонами в комп’ютерних мережах. Версія TLS 1.2, яка є поширеним стандартом у промислових і побутових інформаційних системах, використовує комбінацію симетричного і асиметричного шифрування, хеш-функцій і цифрових сертифікатів для встановлення надійного каналу зв’язку. На початковому етапі відбувається TLS-handshake – процес, у ході якого сторони погоджують криптографічні параметри сесії та здійснюють аутентифікацію. Після цього основна передача даних відбувається з використанням симетричного шифрування, що забезпечує високу швидкість обробки.

BearSSL – це легковажна, портативна і орієнтована на вбудовані пристрої реалізація TLS-бібліотеки, написана мовою C. Її архітектура оптимізована для систем з обмеженими обчислювальними ресурсами, зокрема мікроконтролерів, завдяки чому вона має малий розмір виконуваного коду та мінімальне споживання пам’яті. BearSSL підтримує основні криптографічні алгоритми,

включно з RSA, ECDSA, AES, SHA-2, та протокол TLS 1.2. Однією з характерних рис бібліотеки є можливість роботи з самопідписаними сертифікатами, що дозволяє створювати ізольовані захищені системи без потреби в сертифікаційних центрах.

X.509 – це загальноприйнятий стандарт для форматування цифрових сертифікатів, що використовується в інфраструктурі відкритих ключів (PKI) та протоколах захищеного з'єднання, таких як TLS. Сертифікат X.509 містить відкритий ключ суб'єкта, ідентифікаційні дані (зокрема доменне ім'я або ідентифікатор пристрою), а також цифровий підпис видавця (сертифікаційного центру або самого суб'єкта у випадку самопідписаного сертифіката). Сертифікати X.509 підтримують алгоритми шифрування RSA, ECDSA тощо, а також хеш-функції для підпису, зокрема SHA-256. Самопідписаний X.509-сертифікат, хоч і не підтверджується зовнішнім центром сертифікації, може бути використаний у локальних або контрольованих середовищах для забезпечення автентифікації та встановлення захищеного каналу.

Клієнтський інтерфейс ініціює запит геолокаційних даних через браузерний API navigator.geolocation.getCurrentPosition(), що передбачає використання протоколу W3C Geolocation API. У разі відмови користувача або технічної неможливості визначення позиції, система автоматично активує резервний режим з використанням апаратного GPS-модуля NEO-6M через асинхронний UART-інтерфейс (SoftwareSerial на GPIO D1/D2).

Серверна частина з ендпоїнтом /api/coordinates повертає дані у форматі JSON з полями latitude та longitude, отриманими через парсинг NMEA-рядків бібліотекою TinyGPS++.

Взаємодія з геопросторовими сервісами здійснюється через інтеграцію з OpenStreetMap API (TileLayer) та Nominatim Geocoding Service для реверсного геокодування координат. Візуалізація даних відбувається за допомогою бібліотеки Leaflet.js з використанням асинхронного запиту векторних тайлів через WebSocket-з'єднання.

Для підвищення надійності використано подвійний стек мережевих протоколів: HTTP/HTTPS з автоматичним перенаправленням на захищений канал (код відповіді 301 Moved Permanently). Стан системи моніториться через вбудований mDNS-клієнт із підтримкою multicast DNS (.local домени).

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ ВИЗНАЧЕННЯ МІСЦЕЗНАХОДЖЕННЯ

3.1 Розробка веб-інтерфейсу користувача

Веб-інтерфейс системи GPS-трекінгу реалізований з використанням комбінації HTML5, CSS3 та JavaScript, що забезпечує кросс-платформну сумісність та адаптивність. Архітектура сторінки базується на принципах компонентної модульності, де кожен функціональний блок (мапа, координати, інформація про автора) інкапсульований у власний контейнер з чітко визначеними зонами відповідальності.

Структура DOM-дерева оптимізована для послідовного рендерингу – спочатку завантажується індикатор стану, потім відбувається асинхронна ініціалізація основних компонентів (рис. 3.1).

```
<main>
    <div class="loader">
        // ...
    </div>
    <div class="app">
        <div class="map-container">
            <div id="map"></div>
        </div>
        <div class="info">
            // ...
        </div>
    </div>
</main>
<script
    async function showData(latitude, longitude) {
        // ...
        let mapContainer = document.querySelector('.map-container');
        mapContainer.style.opacity = 1;
        let infoContainer = document.querySelector('.info');
        infoContainer.style.opacity = 1;

        let loader = document.querySelector('.loader');
        loader.style.display = "none";
    }
    // ...
</script>
```

Рисунок 3.1 – Асинхронна ініціалізація основних компонентів

Геометричне позиціонування елементів реалізоване через CSS Flexbox та Grid Layout, що дозволяє динамічно адаптувати інтерфейс під різні роздільні здатності екранів. Для мобільних пристроїв застосовано медіа-запити з умовою

max-width: 960px, які змінюють орієнтацію контейнерів з горизонтальної на вертикальну (рис. 3.2).

```
<head>
    <style>
        .app {
            display: flex;
            flex-direction: row;
        }
        @media (max-width: 960px) {
            .app {
                flex-direction: column;
            }
        }
    </style>
</head>
```

Рисунок 3.2 – Медіа запити для адаптивності інтерфейсу

Механізм оновлення даних використовує гібридний підхід: первинне отримання координат відбувається через Geolocation API браузера, при збоях активується резервний запит до REST API. Асинхронна обробка подій реалізована через Promise chain з використанням async/await для гарантії послідовного виконання операцій. Візуалізація геопозиції на мапі забезпечується бібліотекою Leaflet.js з лінивою ініціалізацією – об'єкт мапи створюється лише після успішного отримання перших координат. Ініціалізація мапи виконується методом L.map('map').setView() з початковим масштабуванням 16x. Візуальна складова забезпечується тайлами OpenStreetMap, які підвантажуються через шаблонний URL https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png (рис. 3.3).

```
// Ініціалізація мапи
map = L.map('map').setView([latitude, longitude], 16);
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {}).addTo(map);
```

Рисунок 3.3 – Лістинг коду для ініціалізації мапи

Стилізація інтерфейсу базується на принципах візуальної ієрархії – ключові елементи (адреса, координати) мають підвищений розмір шрифту та

контрастне форматування. Анімаційні переходи opacity від 0 до 1 використані для плавного появи компонентів після завантаження даних. Стани інтерактивних елементів (наприклад, кнопки оновлення) стилізовані через псевдокласи :hover та :active з кольоровими переходами тривалістю 200ms.

Система обробки помилок реалізує багаторівневу інформаційну підтримку: при відмові у доступі до геолокації відбувається автоматичне переключення на резервне джерело даних з відповідним інформаційним сповіщенням (рис. 3.4). Кольорова семантика (червоний для критичних помилок, сірий для інформаційних повідомлень) сприяє швидкій інтерпретації стану системи.

```
function fetchData() {
    fetch('/api/coordinates')
        .then(response => response.json())
        .then(async coordinates => {
            await showData(coordinates.latitude, coordinates.longitude)
        })
        .catch(error => console.error('Error:', error));
}
function showError(error) {
    // ...
    // Код для виведення помилки
    // ...
    fetchData();
}
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition, showError);
} else {
    document.getElementById('location').innerHTML = "Геолокація не підтримується цим браузером.";
    fetchData()
}
```

Рисунок 3.4 – Лістинг коду для автоматичного перемикання на резервне джерело даних

Як було сказано попередньо, обробка геолокаційних даних здійснюється двома взаємодоповнюючими методами. Первинний механізм використовує браузерну Geolocation API через navigator.geolocation.getCurrentPosition(), яка надає точність до 2.5 метрів у міських умовах. У разі відмови користувача або технічних збоїв активується резервний режим з використанням GPS-модуля NEO-6M через REST API ендпоїнт /api/coordinates.

Інформаційна панель відображає координати з точністю до 6 знаків після коми та зворотний геокод через OpenStreetMap Nominatim API. Асинхронні запити реалізовані через Fetch API з обробкою помилок мережевого рівня.

Оптимізація продуктивності досягнута через кешування DOM-вузлів, мінімізацію перефарбовування сторінки та використання CDN для зовнішніх бібліотек. Точність відображення координат забезпечується методом toFixed(6), що гарантує відображення значень з точністю до 6 знаків після коми, що відповідає вимогам до систем GPS-моніторингу.

3.2 Моделювання електричної схеми

Мікроконтролерна плата WeMos D1 mini, заснована на чипі ESP8266, є компактним та функціональним рішенням для розробки пристроїв Інтернету речей (IoT). Вона поєднує в собі невеликі розміри з широкими можливостями вводу/виводу, що робить її привабливою для різноманітних проектів.

Плата оснащена 11 цифровими портами вводу/виводу (GPIO), які можуть бути налаштовані для виконання різноманітних функцій, включаючи цифровий ввід/вивід, підтримку протоколів I2C, SPI, PWM та інших (рис. 3.5). Крім того, вона має один аналоговий вхід (A0), який дозволяє вимірювати напругу в діапазоні до 3.2 В.

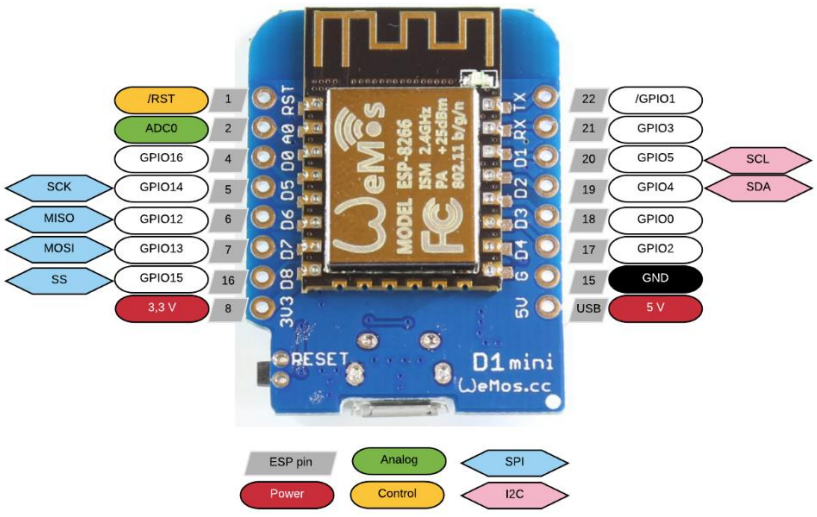

Рисунок 3.5 – Розпіновка Wemos D1 Mini [20]

Кожен цифровий пін має своє позначення (наприклад, D0, D1, D2 тощо) та відповідає певному GPIO на чипі ESP8266. Наприклад, пін D1 відповідає GPIO5 і часто використовується як лінія SCL для I2C, тоді як D2 відповідає GPIO4 і зазвичай використовується як SDA. Деякі піни мають специфічні особливості, які слід враховувати при проектуванні схем. Наприклад, пін D3 (GPIO0) має підтягувальний резистор до високого рівня і пов'язаний з кнопкою завантаження прошивки; якщо він буде підтягнутий до низького рівня під час запуску, це може призвести до входу в режим програмування. Аналогічно, пін D4 (GPIO2) також має підтягувальний резистор і пов'язаний з вбудованим світлодіодом; його стан під час запуску може впливати на процес завантаження.

Модуль GPS NEO-6M є компактним та енергоефективним пристроєм, призначеним для визначення географічного положення за допомогою супутникових сигналів. Він широко використовується в проектах на базі мікроконтролерів, таких як Arduino та ESP8266, завдяки простоті інтеграції та надійності (рис. 3.6).

Рисунок 3.6 – Розпіновка NEO-6M [21]

Пін VCC призначений для подачі живлення на модуль. NEO-6M підтримує вхідну напругу в діапазоні від 3.3 В до 5 В, що дозволяє підключати його

безпосередньо до відповідних виходів мікроконтролера без необхідності додаткових стабілізаторів напруги.

Пін GND використовується для заземлення модуля. Цей пін слід з'єднати з загальним "мінусом" системи, забезпечуючи спільну точку відліку напруги для всіх компонентів.

Пін TX (Transmitter) використовується для передачі даних. NEO-6M передає дані про місцезнаходження через цей пін у форматі NMEA з типовою швидкістю 9600 бод. Його слід підключити до приймального піна (RX) мікроконтролера для отримання інформації.

Пін RX (Receiver) використовується для прийому даних. Через цей пін модуль може приймати команди або конфігураційні дані від мікроконтролера. Оскільки NEO-6M працює з логічними рівнями 3.3 В, при підключенні до мікроконтролерів, що використовують 5 В логіку, рекомендується використовувати дільник напруги або логічний рівневий перетворювач для захисту модуля від перенапруги.

Додатково, деякі версії модуля можуть мати пін PPS (Pulse Per Second), який генерує імпульси з високою точністю, що може бути корисним для синхронізації часу в деяких додатках.

Для забезпечення надійного прийому супутникових сигналів, модуль оснащений керамічною антеною. У випадках, коли вбудована антена не забезпечує достатнього рівня сигналу, можливо підключення зовнішньої антени через відповідний роз'єм.

Завдяки вбудованій енергонезалежній пам'яті EEPROM та резервному живленню від батареї, модуль може зберігати останні налаштування та інформацію про супутники, що дозволяє скоротити час до першого визначення координат після включення.

Таким чином, NEO-6M є ефективним рішенням для реалізації функцій геолокації в різноманітних електронних проектах, забезпечуючи точне та стабільне визначення місцезнаходження.

Підключення модуля GPS NEO-6M до мікроконтролерної плати WeMos D1 mini здійснюється через інтерфейс UART, використовуючи цифрові порти D1 та D2. Цей процес забезпечує надійний обмін даними між пристроями та дозволяє отримувати точну інформацію про географічне положення.

Для передачі даних використовується пін TX модуля, який підключається до порту D2 (GPIO4) на платі WeMos D1 mini (рис. 3.7). Цей пін відповідає за надсилання даних від модуля до мікроконтролера. Пін RX модуля, призначений для прийому даних, підключається до порту D1 (GPIO5) на платі.

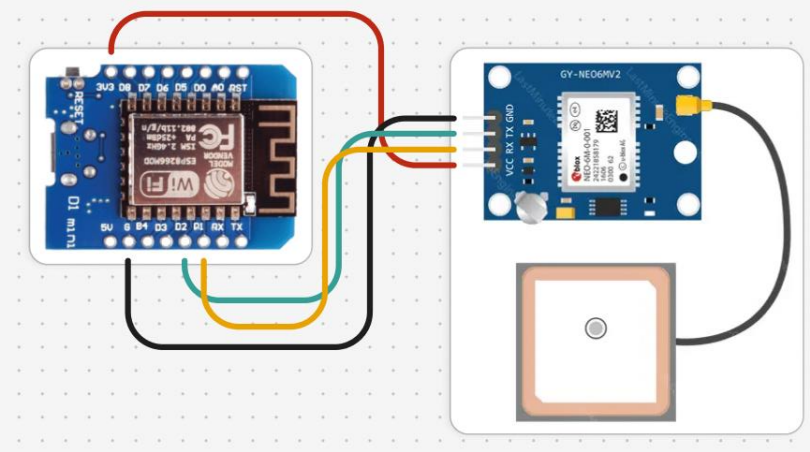

Рисунок 3.7 – Електрична схема розробки

Фізична електрична схема, спроектована на макетній платі, зображена на рисунку 3.8.

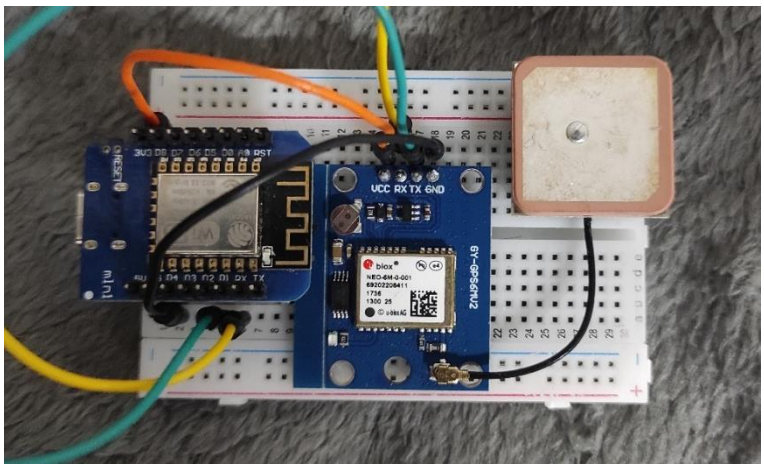

Рисунок 3.8 – Фізична електрична схема

Варто також зазначити, що оскільки мікроконтролер працює з логічними рівнями 3.3 В, а модуль NEO-6M може бути живлений від 5 В, рекомендується використовувати дільник напруги або логічний рівневий перетворювач для зниження напруги на піні RX модуля до безпечного рівня.

3.3 Програмування мікроконтролеру Wemos D1 Mini

Програмна реалізація мікроконтролера базується на архітектурі подійно-орієнтованого програмування з використанням бібліотек ESP8266WebServerSecure та TinyGPS++. Ініціалізація апаратних інтерфейсів виконується у функції setup(), де первинно налаштовується UART-з'єднання з GPS-модулем через програмну емуляцію послідовного порту на виводах D1 та D2.

Для забезпечення безпечного зв'язку реалізовано двонаправлену комунікацію: HTTP-сервер на порту 80 виконує перенаправлення на HTTPS-ендпоїнт, а захищений сервер на порту 443 обробляє TLS-з'єднання з використанням самопідписаного X.509 сертифіката.

Генерація криптографічних ключів виконується у терміналі за допомогою OpenSSL-команди, зображеної на рисунку 3.9.

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.pem
```

Рисунок 3.9 – Електрична схема розробки

Створені PEM-файли конвертуються у PROGMEM-рядки для інтеграції у прошивку. Статичні константи serverCert та serverKey ініціалізують SSL-контекст сервера методом setRSACert(), що забезпечує 128-бітне шифрування TLS 1.2. Приклад PROGMEM-рядків зображено на рисунку 3.10.

```
static const char serverCert[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIC6jCCAlOgAwIBAgIUCMsOb90tv3Bo7V0mHtUK3skD2xcwDQYJKoZIhvcNAQEL
...
-----END CERTIFICATE-----
)EOF";

static const char serverKey[] PROGMEM = R"EOF(
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBANe1//zHAG7dVJKT
...
-----END PRIVATE KEY-----
)EOF";
```

Рисунок 3.10 – Приклад PROGEM-рядків

У функції setup() виконується прив'язка маршруту /api/coordinates до HTTP GET-запиту, обробку якого здійснює функція handleCoordinates (рис. 3.11). Ця функція формує JSON-об’єкт із даними про широту та довготу, отриманими з GPS-модуля за допомогою методу gps.location.lat() і gps.location.lng(), де точність обмежена шістьма знаками після коми.

```
// Обробник REST API для GPS-даних
void handleCoordinates() {
  String jsonResponse = "{ \"latitude\": " + String(gps.location.lat(),6) + ", \"longitude\": ";
  jsonResponse += String(gps.location.lng(),6) + " }";
  server.send(200, "application/json", jsonResponse);
}

// Ініціалізація захищеного сервера
BearSSL::ESP8266WebServerSecure server(443);
void setup() {
  // ...
  server.getServer().setRSACert(new BearSSL::X509List(serverCert), new BearSSL::PrivateKey(serverKey));
  serverHTTP.on("/api/coordinates", HTTP_GET, handleCoordinates);
  // ...
}
```

Рисунок 3.11 – Лістинг коду функції handleCoordinates

Згенерований рядок у форматі JSON надсилається клієнту у відповідь на запит з HTTP-кодом 200 та вказаним MIME-типом application/json. Таким чином, система надає клієнтам доступ до актуальних координат у захищеному форматі через HTTPS API-інтерфейс.

Обробка GPS-даних реалізована через циклічне опитування буфера UART у функції loop() (рис. 3.12). Бібліотека TinyGPS++ декодує NMEA-фрейми типу

GGA та RMC, екстрагуючи геодезичні координати з точністю до 6 знаків після коми. Валідація отриманих даних здійснюється через метод location.isValid(), який перевіряє актуальність та цілісність навігаційних параметрів.

```
TinyGPSPlus gps;

void loop() {
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());  // Зчитування даних з GPS-модуля
  }

  server.handleClient();
  MDNS.update();
}
```

Рисунок 3.12 – Лістинг коду функції loop

Мережева інфраструктура включає двоетапне підключення: первинна автентифікація у WiFi-мережі з використанням WPA2-PSK, після чого активується mDNS-резолвінг для доступу через доменне ім'я geolocation.local. У разі збою мережевого підключення передбачено апаратний ресет через 60 секунд неактивності.

3.4 Тестування працездатності системи

Для початку роботи з системою, першим кроком необхідно створити точку доступу з назвою «wemos-geo-tracker (рис. 3.13), після чого мікроконтролер автоматично підключиться до неї.

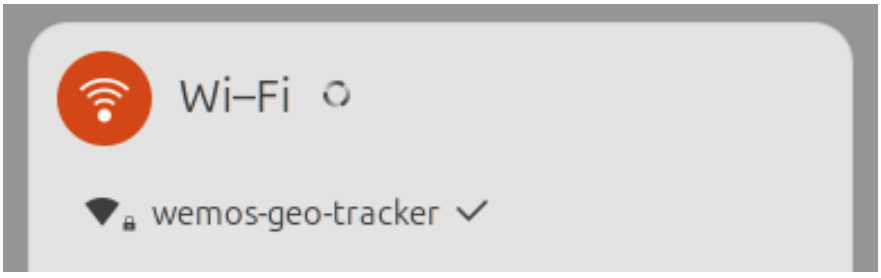

Рисунок 3.13 – Підключення до точки доступу

Перевірити підключення мікроконтролеру можна через будь який мережевий сканер. Для прикладу, на рисунку 3.14 зображено пошук мікроконтролеру в локальній мережі через nmap.

```
nmap -sn 192.168.0.1/24
Starting Nmap 7.94SVN
Nmap scan report for _gateway (192.168.0.1)
Host is up (0.0100s latency).
Nmap scan report for ESP_60C3F8 (192.168.34.1)
Host is up (0.0100s latency).
```

Рисунок 3.14 – Сканування мережі через nmap

Так як у при розробці на wemos-сервері було налаштоване MDNS, отримати доступ до сторінки можна перейшовши за адресою https://geolocation.local, або ж, як альтернативний варіант, можна підключитися напряму через IP адресу. При відкритті сторінки браузер запитає користувача дозвіл на доступ до геолокації пристрою (рис. 3.15).

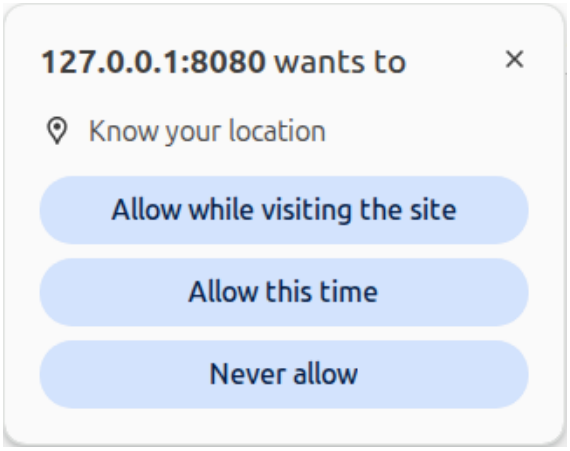

Рисунок 3.15 – Сповіщення з запитом на доступ до геолокації

Якщо користувач заблокує доступ, то на сторінці з’явиться відповідний інформер (рис. 3.16) і система переключиться на роботу від NEO-6M та через певний час відобразиться карта з геолокацією.

```
Користувач відмовився надати доступ до геолокації.
Визначення геолокації за допомогою NEO-6M...
```

Рисунок 3.16 – Сповіщення з запитом на доступ до геолокації

Якщо ж користувач надасть браузеру доступ до геолокації, то карта з координатами відобразиться одразу (рис. 3.17).

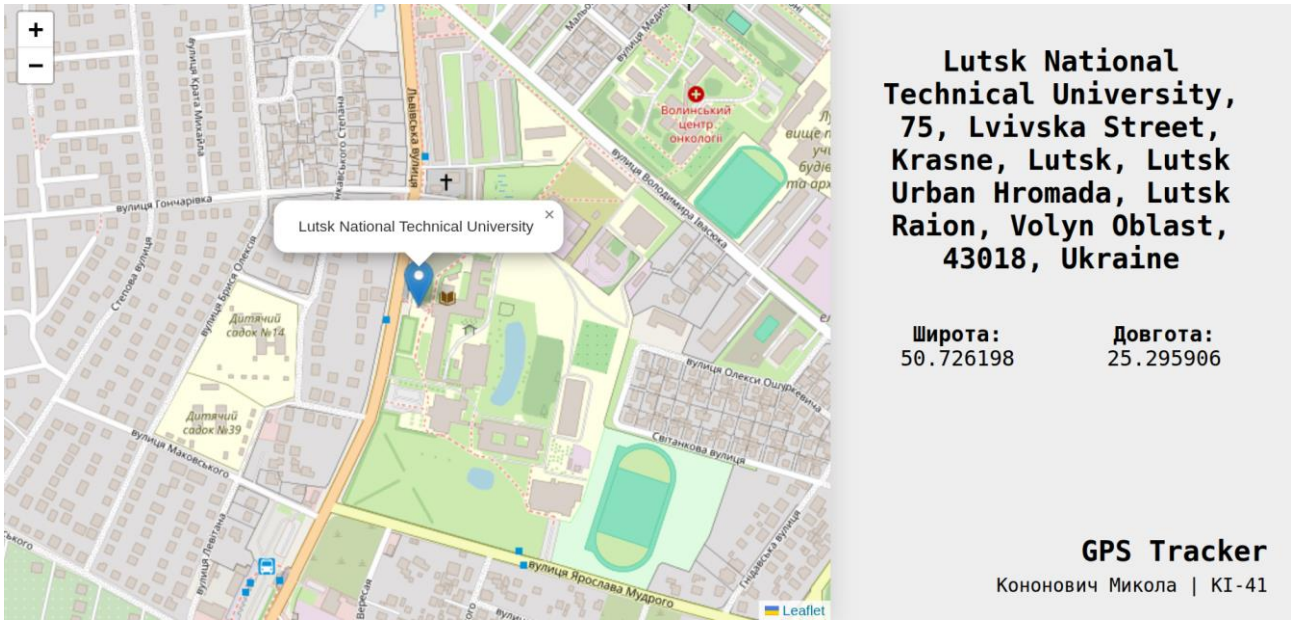

Рисунок 3.17 – Карта з геолокацією користувача

Практичні вимірювання точності позиціювання виявили статистично значущу різницю між джерелами даних. Браузерний API демонстрував середньоквадратичне відхилення 2.8 метра у міських умовах, тоді як апаратний GPS-модуль мав похибку 4.2 метра при роботі під відкритим небом. Це обумовлено обмеженням кількості супутників, доступних для аналізу в умовах міської забудови.

При використанні системи з мобільного пристрою, сторінка автоматично адаптується до необхідних розмірів екрану, запезпечуючи коректне відображення інформації на портативних пристроях (рис. 3.18).

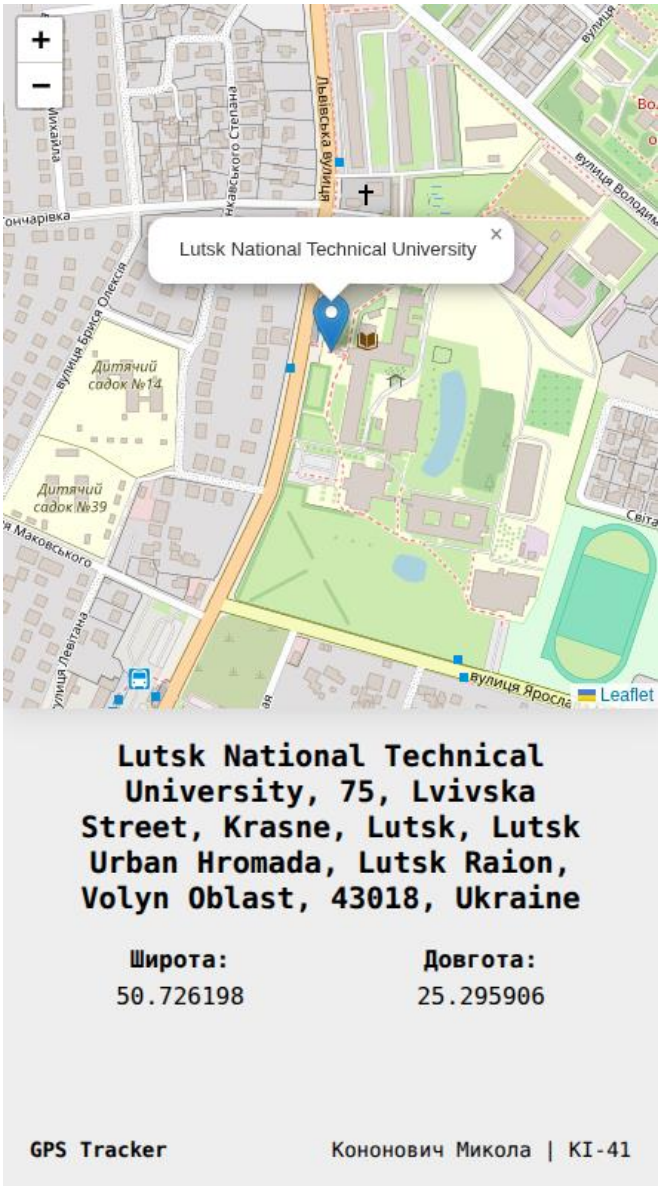

Рисунок 3.18 – Мобільна версія сторінки

ВИСНОВКИ

У ході роботи було досліджено основні теоретичні відомості про мікроконтролерні системи та здійснено комплексний аналіз з порівнянням архітектурних особливостей ESP8266, Arduino Uno та STM32. Встановлено переваги Wemos D1 Mini у контексті IoT-рішень: інтеграція Wi-Fi модуля, 4 МБ флеш-пам'яті та підтримка UART/I2C інтерфейсів для підключення периферії.

Було спроектовано клієнт-серверну архітектуру з використанням двонаправленого TLS 1.2 зв'язку, де мікроконтролер виконує роль HTTPS-сервера з підтримкою REST API. Реалізовано механізм mDNS-резолвінгу для доступу через локальний домен та автоматичне перенаправлення HTTP→HTTPS із кодом 301.

Наступним кроком було реалізовано гібридний механізм визначення геолокації з перемиканням між браузерним API (точність 2.5 м) та апаратним GPS NEO-6M (точність до 5 м). Використано подієво-орієнтоване програмування з бібліотекою TinyGPS++ для аналізу NMEA-даних від 14 супутників.

Для роботи системи було організовано захищений веб-інтерфейс з підтримкою HTTPS/TLS 1.2 та REST API. Для цього було згенеровано SSL сертифікат, який потім було конвертовано в PROGMEM-змінні, що дозволило використовувати захищене HTTPS з'єднання з Wemos-сервером.

Було також розроблено адаптивний веб-інтерфейс з інтеграцією Leaflet.js 1.9.4 та OpenStreetMap API, що забезпечує візуалізацію координат у форматі JSON через захищений ендпоїнт /api/coordinates. Інтерфейс адаптується до будь-яких розмірів екрану, що відповідає сучасним вимогам до веб-інтерфейсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to Microcontrollers. URL: https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller (дата звернення: 20.02.2024).

2. Гетьман К. Р., Шевцов І. О. Універсальний програмно-апаратний інтерфейс зв'язку з множиною вбудованих пристроїв // Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології : матеріали IV форуму, 24–25 листопада 2022 р. Харків : ХНУРЕ, 2022. С. 87–88.

3. What is a microcontroller (MCU)? URL: https://amn.com.sa/what-is-a-microcontroller-mcu/ (дата звернення: 22.02.2024).

4. Татарчук Д. Д., Діденко Ю. В. Мікропроцесори та мікроконтролери. Курс лекцій: навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2020. 238 с.

5. How to program a microcontroller? Top microcontroller programming languages. URL: https://www.tme.eu/en/news/library-articles/page/58200/how-to-program-a-microcontroller-top-microcontroller-programming-languages/ (дата звернення: 02.03.2024).

6. Wemos D1 mini module (ESP-8266). URL: https://www.researchgate.net/figure/Wemos-D1-mini-module-ESP-8266_fig2_333686151 (дата звернення: 05.03.2024).

7. MICRO/ Type C ESP8266 ESP-12 CH340G CH340. URL: https://www.alibaba.com/product-detail/MICRO-Type-C-ESP8266-ESP-12_1600051643483.html (дата звернення: 10.03.2024).

8. Зубков О. В., Зубков А. О. Особливості реалізації Web серверів на модулях ESP8266 ТА ESP32 У Arduino IDE // IV форум «Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології» AERT-2022. Харків, ХНУРЕ, 2022. С. 54–55.

9. Осів С. Використання мови програмування Micropython на мікроконтролерах // Матеріали VII Міжнародної студентської науково-технічної

конференції «Природничі та гуманітарні науки. Актуальні питання», 25-26 квітня 2024 року. Т. : ТНТУ, 2024. С. 9–10.

10. Comparison microcontroller performance. URL: https://mischianti.org/comparison-microcontroller-performance/ (дата звернення: 15.03.2024).

11. Adafruit GPS Library. URL: https://docs.arduino.cc/libraries/adafruit-gps-library/ (дата звернення: 23.03.2024).

12. Все про всенаправлені Wi-Fi антени. URL: https://infotabula.com/elektronika/vse-pro-vsenapravleni-wi-fi-anteni.html (дата звернення: 02.04.2024).

13. Здолбіцька Н. В., Мельник Г. В., Мельник В. М., Колтунович О. С., Мазуренко В. В. Аналіз роботи послідовного протоколу UART за допомогою цифрового аналізатора. Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». Луцьк, 2020. Випуск № 41. С. 165–173.

14. HTML Geolocation API. URL: https://www.w3schools.com/htmL/html5_geolocation.asp (дата звернення: 08.04.2024).

15. GPS-приймач GY-GPS6MV2. URL: https://www.mini-tech.com.ua/ua/gps-priemnik-gy-gps6mv2 (дата звернення: 14.04.2024).

16. Зубрєва Д. Ю. Дослідження методів та засобів розробки веб-застосувань для програмної реалізації веб-сервісу щодо проведення міських квестів. Харків : НУР, 2021. 59 с.

17. REST interface in Industrial IoT. URL: https://i-flow.io/en/ressources/connect-almost-any-system-in-ot-and-it-with-rest/ (дата звернення: 20.04.2024).

18. Маренич В. В. Дослідження методів розробки комп'ютерних застосунків з інтегрованою API. Харків : НУР, 2025. 64 с.

19. Yevseiev, S., Havrylova, A., Milevskyi, S., Sinitsyn, I., Chalapko, V., Dukin, H., Hrebeniuk, V., Diedov, M.,Bekirova, L., Shpak, O. (2023). Development of an improved ssl/tls protocol using post-quantum algorithms. Eastern-Euro-pean Journal of Enterprise Technologies, 3 (9 (123)). P. 33–48.

20. WeMos D1 mini pins and diagram. URL: https://escapequotes.net/esp8266-wemos-d1-mini-pins-and-diagram/ (дата звернення: 28.04.2024).

21. Interface ublox NEO-6M GPS Module. URL: https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/ (дата звернення: 02.05.2024).

ДОДАТКИ

Додаток А

Лістинг коду прошивки Wemos D1 Mini

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h>
#include <ESP8266WebServerSecure.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include "page.h"

SoftwareSerial gpsSerial(D1, D2);
TinyGPSPlus gps;

const char *ssid = "wemos-geo-tracker";
const char *password = "wemosGeoTracker123";
const char *dname = "geolocation";

BearSSL::ESP8266WebServerSecure server(443);
ESP8266WebServer serverHTTP(80);

static const char serverCert[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIC6jCCAlOgAwIBAgIUCMsOb90tv3Bo7V0mHtUK3skD2xcwDQYJKoZIhvcNAQEL
BQAwejELMAkGA1UEBhMCUk8xCjAIBgNVBAgMAUIxEjAQBgNVBAcMCUJ1Y2hhcmVz
dDEbMBkGA1UECgwST25lVHJhbnNpc3RvciBbUk9dMRYwFAYDVQQLDA1PbmVUcmFu
c2lzdG9yMRYwFAYDVQQDDA1lc3A4MjY2LmxvY2FsMB4XDTI1MDMxOTE3Mzg1NloX
DTI2MDMxOTE3Mzg1NlowejELMAkGA1UEBhMCUk8xCjAIBgNVBAgMAUIxEjAQBgNV
BAcMCUJ1Y2hhcmVzdDEbMBkGA1UECgwST25lVHJhbnNpc3RvciBbUk9dMRYwFAYD
VQQLDA1PbmVUcmFuc2lzdG9yMRYwFAYDVQQDDA1lc3A4MjY2LmxvY2FsMIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXtf/8xwBu3VSSk4kD238mlgmZzs9mhPU5
qT4GejopxpSoHOtbhsV7PouB2PvpyfLpyzi+dLGoRvxj2OmA3k+tifDAcvGobjrS
HUx/WCL/ioGaRTlJlE8nOrxvKMEpG+FAVSZZi9Sk/oJGBsoR7H77/ONQX4fuSgjw
oKvLWD9f0wIDAQABo20wazAdBgNVHQ4EFgQUa8WZAjEwwKrvDTMDoDtJgFOpLy4w
HwYDVR0jBBgwFoAUa8WZAjEwwKrvDTMDoDtJgFOpLy4wDwYDVR0TAQH/BAUwAwEB
/zAYBgNVHREEETAPgg1lc3A4MjY2LmxvY2FsMA0GCSqGSIb3DQEBCwUAA4GBAD/+
LTYCVQnsNzn1Jdd1Y/14k/VV8VPUnWwszsUORTUAMnTtMnL+i5HZzAPqwwamWUK2
BghhHedvNROSexlabZ05Tii0DVq9CKmFbbHF8fe6on8IjghfR1iAchN33+NXrslr
AUcdihcu17ExdtRyP73JUbNYCHz4fhW2vrTqfQFm
-----END CERTIFICATE-----
)EOF";

static const char serverKey[] PROGMEM = R"EOF(
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBANe1//zHAG7dVJKT
iQPbfyaWCZnOz2aE9TmpPgZ6OinGlKgc61uGxXs+i4HY++nJ8unLOL50sahG/GPY
6YDeT62J8MBy8ahuOtIdTH9YIv+KgZpFOUmUTyc6vG8owSkb4UBVJlmL1KT+gkYG
yhHsfvv841Bfh+5KCPCgq8tYP1/TAgMBAAECgYBhy5GS+GWp3Y5KJbkoloq+46bW
pHxC7mR/D8ufBCr6ZZ1f2jyZ8i/2ABUfeVA8XeJTPF8teZpRwF0Mp3q1ghR8s7g0
FpyaF7KWDdz/2SSLntx2gpgxLy8gUWQb4iijWkeRepMzVmagNwKv8CJssyUFB/HK
t9fLXXCto+0s2mLcAQJBAPDrNeBDYDULUDAS2z4n3/Sghxcvg2TA9NqrfUp1mBrB
N2B0XlNmRIfdZMM95fuimulZQ/CpfOVL4ef+NO9YWeECQQDlNtOlsvaTPBJljDyn
74CbdjCNK6eEnEMwd6vrxCTC/tkZ4FG2NWKNOkG9QTRwo6ZOe7BDAkIQRDMTbrfY
BHgzAkA1hCo9nPncDp1LcITfM8Bd+OLJVbacRPifMoTsa17UYqNIe9EQOaFzhMyD
SlayJRbASyH6bcveq5KNFDzyBfBBAkBqJRqb2CVhh2cJiYipSDrpQzE2nqmjum1O
64w80QcpGbVVy1BrrYsm1XyN9L0hEFe6boLwdQI84Cg67oHKUj+LAkEAlrJZ6tfY
jyp7ZOBL0QKXxR7CEK4Q2bn0tKcc/9KoXGsFxi8LJ5FAtAvSFpeFJ+nZ/s5xIfYN
A+maVjq1TxmGmg==
-----END PRIVATE KEY-----
)EOF";
```

```
bool connectToWifi()
{
  byte timeout = 50;

  Serial.println("\n\n");

  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  for (int i = 0; i < timeout; i++)
  {
    if (WiFi.status() == WL_CONNECTED)
    {
      Serial.println("\nConnected to WiFi");
      Serial.print("Server can be accessed at https://");
      Serial.print(WiFi.localIP());
      if (MDNS.begin(dname))
      {
        Serial.print(" or at https://");
        Serial.print(dname);
        Serial.println(".local");
      }
      return true;
    }
    delay(5000);
    Serial.print(".");
  }

  Serial.println("\nFailed to connect to WiFi");
  Serial.println("Check network status and access data");
  Serial.println("Push RST to try again");
  return false;
}
void handleRoot()
{
  server.send(200, "text/html", PAGE_MAIN);
}

void secureRedirect()
{
  serverHTTP.sendHeader("Location", String("https://geolocation.local"), true);
  serverHTTP.send(301, "text/plain", "");
}

void handleCoordinates()
{
  String jsonResponse = "{}";

  if (gps.location.isValid())
  {
    jsonResponse = "{ \"latitude\": " + String(gps.location.lat(), 6) + ",
\"longitude\": " + String(gps.location.lng(), 6) + " }";
  }
  else
  {
    jsonResponse = "{ \"error\": \"No GPS data available\" }";
  }

  server.send(200, "application/json", jsonResponse);
}

void setup()
```

```
{
  pinMode(D0, OUTPUT);
  Serial.begin(115200);

  if (!connectToWifi())
  {
    delay(60000);
    ESP.restart();
  }

  serverHTTP.on("/", secureRedirect);
  serverHTTP.on("/api/coordinates", HTTP_GET, handleCoordinates);
  serverHTTP.begin();

  server.getServer().setRSACert(new BearSSL::X509List(serverCert), new
BearSSL::PrivateKey(serverKey));
  server.on("/", handleRoot);
  server.begin();

  Serial.println("Server is ready");
}

void loop()
{
  while (gpsSerial.available() > 0)
  {
    gps.encode(gpsSerial.read()); // Зчитуємо дані з GPS-модуля
  }

  serverHTTP.handleClient();
  server.handleClient();
  MDNS.update();
}
```

Додаток Б

Лістинг коду веб-інтерфейсу

```
const char PAGE_MAIN[] PROGMEM = R"=====(
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>GPS Tracker</title>
    <meta
            name="viewport"
            content="width=device-width, initial-scale=1.0"
    >
    <link
            rel="stylesheet"
            href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
    />
    <style>
        * {
            margin: 0;
        }

        html, body {
            height: 100%;
        }

        body {
            font-family: monospace;
            background-color: #eeeeee;
            display: flex;
            flex-direction: column;
            overflow: hidden;
        }

        h1 {
            font-size: 24px;
            font-weight: bold;
        }

        .author {
            margin-top: auto;
            text-align: right;
        }

        .name {
            font-size: 16px;
            margin-top: 6px;
        }

        .map-container {
            opacity: 0;
            flex: 2;
            width: 100%;
            z-index: 1;
            height: 100%;
            box-shadow: 4px 4px 20px rgba(0, 0, 0, 0.1);
        }

        #map {
            width: 100%;
            height: 100%;
```

```
}

main {
    height: 100%;
}

.app {
    display: flex;
    flex-direction: row;
    height: 100%;
}

.info {
    opacity: 0;
    flex: 1;
    padding: 20px;
    box-sizing: content-box;
    flex-direction: column;
    display: flex;
    row-gap: 20px;
}

.update {
    background-color: transparent;
    padding: 16px;
    font-size: 18px;
    border: 1px solid black;
    width: 100%;
    cursor: pointer;
    transition: all 0.2s ease;
    font-weight: 600;
    font-family: monospace;
}

.update:hover {
    background-color: black;
    color: white;
}

.update:active {
    background-color: white !important;
    color: black;
}

.coordinates {
    display: grid;
    grid-template-columns: 1fr 1fr;
    font-size: 18px;
}

.coordinates * {
    text-align: center;
}

.address {
    text-align: center;
    font-weight: bold;
    font-size: 24px;
    margin: 20px;
}

.loader {
    height: calc(100%);
    display: flex;
```

```
            align-items: center;
            justify-content: center;
            font-size: 24px;
            color: #5a5a5a;
            font-weight: 600;
            text-align: center;
            flex-direction: column;
        }

        #location {
            position: absolute;
            top: 0;
            left: 0;
        }

        @media (max-width: 960px) {
            .app {
                flex-direction: column;
            }

            .map-container {
                flex: 5;
            }

            .info {
                flex: 3;
            }

            .update {
                font-size: 16px;
            }

            .coordinates {
                font-size: 16px;
            }

            .coordinates b {
                display: block;
                margin-bottom: 6px;
            }

            h1 {
                font-size: 14px;
            }

            .author {
                display: flex;
                align-items: center;
                justify-content: space-between;
            }

            .name {
                font-size: 14px;
                margin: 0;
            }

            .address {
                margin-bottom: 0;
                margin-top: 0;
                font-size: 20px;
            }
        }
    </style>
</head>
```

```
<body>
<main>
    <div class="loader">
        <span class="loader-title">Завантаження...</span>
        <br/>
        <span class="loader-subtitle"></span>
    </div>
    <div class="app">
        <div class="map-container">
            <div id="map"></div>
        </div>
        <div class="info">
            <p class="address"></p>
            <div class="coordinates">
                <p><b>Широта: </b><span class="js-latitude"></span></p>
                <p><b>Довгота: </b><span class="js-longitude"></span></p>
            </div>
            <div class="author">
                <h1>GPS Tracker</h1>
                <p class='name'>Кононович Микола | КІ-41</p>
            </div>
        </div>
    </div>
</main>
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
<script>
    let map; // Змінна для збереження карти
    let marker; // Змінна для збереження маркера
    let lat, lon;

    async function showData(latitude, longitude) {
        const address = await
fetch(`https://nominatim.openstreetmap.org/reverse?format=json&lat=${latitude}&l
on=${longitude}`).then(res => res.json())

        if (!map) {
            map = L.map('map').setView([latitude, longitude], 16);
            L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
{}).addTo(map);
        } else {
            map.setView([latitude, longitude], 16);
        }

        if (marker) {
            marker.setLatLng([latitude, longitude])
                .bindPopup(address.name.length > 0 ? address.name :
address.address.village)
                .openPopup();
        } else {
            marker = L.marker([latitude, longitude]).addTo(map)
                .bindPopup(address.name.length > 0 ? address.name :
address.address.village)
                .openPopup();
        }

        // Оновлюємо координати на сторінці
        let htmlAddress = document.querySelector('.address');
        htmlAddress.textContent = address.display_name

        let htmlLatitude = document.querySelector('.js-latitude');
        htmlLatitude.textContent = latitude.toFixed(6);
        let htmlLongitude = document.querySelector('.js-longitude');
        htmlLongitude.textContent = longitude.toFixed(6);
```

```
        let mapContainer = document.querySelector('.map-container');
        mapContainer.style.opacity = 1;
        let infoContainer = document.querySelector('.info');
        infoContainer.style.opacity = 1;

        let loader = document.querySelector('.loader');
        loader.style.display = "none";
    }

    function fetchData() {
        fetch('/api/coordinates')
            .then(response => response.json())
            .then(async coordinates => {
                await showData(coordinates.latitude, coordinates.longitude)
            })
            .catch(error => console.error('Error:', error));
    }

    function showPosition(position) {
        lat = position.coords.latitude;
        lon = position.coords.longitude;
        showData(lat, lon).then()
    }

    function showError(error) {
        fetchData()
        const loaderTitle = document.querySelector('.loader-title')
        const loaderSubtitle = document.querySelector('.loader-subtitle')
        switch (error.code) {
            case error.PERMISSION_DENIED:
                loaderTitle.innerHTML = "Користувач відмовився надати доступ до
геолокації.";
                loaderTitle.style.color = "#ee0000";
                loaderSubtitle.innerHTML = "Визначення геолокації за допомогою
NEO-6M...";
                loaderSubtitle.style.color = "#595959";
                break;
            case error.POSITION_UNAVAILABLE:
                loaderTitle.innerHTML = "Інформація про місцезнаходження
недоступна.";
                loaderTitle.style.color = "#ee0000";
                loaderSubtitle.innerHTML = "Визначення геолокації за допомогою
NEO-6M...";
                loaderSubtitle.style.color = "#595959";
                break;
            case error.TIMEOUT:
                loaderTitle.innerHTML = "Час очікування запиту геолокації
вичерпано.";
                loaderTitle.style.color = "#ee0000";
                loaderSubtitle.innerHTML = "Визначення геолокації за допомогою
NEO-6M...";
                loaderSubtitle.style.color = "#595959";
                break;
            case error.UNKNOWN_ERROR:
                loaderTitle.innerHTML = "Сталася невідома помилка.";
                loaderTitle.style.color = "#ee0000";
                loaderSubtitle.innerHTML = "Визначення геолокації за допомогою
NEO-6M...";
                loaderSubtitle.style.color = "#595959";
                break;
        }
    }

    if (navigator.geolocation) {
```

```
        navigator.geolocation.getCurrentPosition(showPosition, showError);
    } else {
        document.getElementById('location').innerHTML = "Геолокація не
підтримується цим браузером.";
    }
</script>
</body>
</html>
)=====";
```